# Partial Reconfiguration of a Processor Peripheral Tutorial

**PlanAhead Design Tool** 

UG744 (v14.1) April 24, 2012





#### **Notice of Disclaimer**

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <u>http://www.xilinx.com/warranty.htm#critapps</u>.

©Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

#### **Revision History**

The following table shows the revision history for this document.

Date	Version	Revision
07/06/2011	13.2	Revalidated for the 13.2 release. Editorial updates only; no technical content updates.
10/19/2011	13.3	Updated the tutorial to use an AXI4-based design.
1/18/2012	13.4	Revalidated for the 13.4 release. Editorial updates only; no technical content updates.
4/24/2012	14.1	Revalidated for the 14.1 release. Updated HWICAP version.

# **Table of Contents**

Partial Reconfiguration of a Processor Peripheral Tutorial	4
Introduction	4
Tutorial Objectives	5
Getting Started	5
Tutorial Steps	8
Step 1: Creating a Processor Hardware System	
Step 2: Creating a Software Project	14
Step 3: Creating a PlanAhead Project	
Step 4: Defining a Reconfigurable Partition	
Step 5: Adding Reconfigurable Modules	24
Step 6: Defining the Reconfigurable Partition Region	27
Step 7: Running the Design Rule Checker	30
Step 8: Creating the First Configuration, Implementing, and Promoting	31
Step 9: Creating Other Configurations, and Implementing	35
Step 10: Running Partial Reconfiguration to Verify Utility	37
Step 11: Generating Bit Files	38
Step 12: Creating an Image, and Testing	
Conclusion	41

# Chapter 1

# Partial Reconfiguration of a Processor Peripheral Tutorial

# Introduction

This tutorial shows you how to develop a partial reconfiguration design using the Xilinx® Platform Studio (XPS), Software Development Kit (SDK), and the PlanAhead<sup>™</sup> design tool. You will use XPS to create a processor hardware system that includes a lower-level module defining one Reconfigurable Partition (RP) and two Reconfigurable Modules (RM). The two RM perform addition and multiplication functions. You will use SDK to create a software application that enables you to perform partial reconfiguration.

XPS and SDK are part of the Embedded Design Kit (EDK), which is included in the ISE® Design Suite Embedded and System Editions.

You will use PlanAhead to:

- Floorplan the design including defining a reconfigurable partition for the reconfigurable region
- Create multiple configurations and run the partial reconfiguration implementation flow to generate full and partial bitstreams.

You will use the ML-605 evaluation board to verify the design in hardware using a Compact Flash (CF) memory card to configure the FPGA device initially and then partially reconfigure the device using the AXI HWICAP peripheral by loading the partial bitstream files stored on the CF under the user software control.

This tutorial covers only a subset of the features contained in the PlanAhead tool bundled with ISE Design Suite Release. Other features are covered in other tutorials.

# **Tutorial Objectives**

After completing this tutorial, you will be able to:

- Generate a processor system using XPS and SDK.
- Use the Partial Reconfiguration design flow capability in PlanAhead to generate full- and partial-bitstreams to dynamically reconfigure an FPGA design using the AXI HWICAP peripheral.

# **Getting Started**

#### **Software Requirements**

The PlanAhead tool installs with the ISE Design Suite software. For this tutorial, you must have the Embedded or System edition of the ISE Design Suite installed. Before starting the tutorial, ensure that the software is operational and the reference design is unzipped and installed.

For PlanAhead installation instructions and information, refer to *the ISE Design Suite 14*: *Installation, and Licensing Guide* on the Xilinx website:

http://www.xilinx.com/support/documentation/sw\_manuals/xilinx14\_1/iil.pdf

You must obtain a FlexLM license for Partial Reconfiguration to access the Partial Reconfiguration features. Contact your Xilinx Field Applications Engineer, or go to the Xilinx website at: <u>http://www.xilinx.com/getproduct</u>

#### **Hardware Requirements**

Xilinx recommends a minimum of 2 GB of RAM for use with this design for best performance.

Optionally, you can use an ML605 board and a USB download cable to test the hardware.

#### Locating the Tutorial Design Files

This tutorial uses a reference design, UG744\_design\_files.zip, which must be unzipped to a directory on your machine. Please note that the directory path you choose should not have a space in its name. You can download a copy of the reference design from the Xilinx website:

http://www.xilinx.com/support/documentation/dt planahead planahead14-1 tutorials.htm



#### **Understanding the Processor System**

This tutorial demonstrates how to implement a design that can be dynamically reconfigured using the AXI HWICAP peripheral.

The following figure shows a processor system. The design consists of a peripheral capable of performing a math function, having two unique capabilities: addition and multiplication.

You will verify the functionality with HyperTerminal under user application control. The dynamic modules are reconfigured using the AXI HWICAP peripheral.

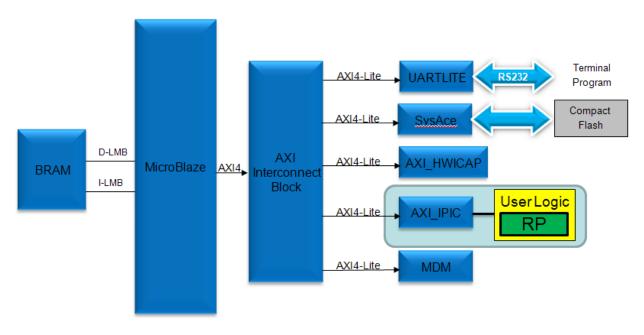


Figure 1: Top-Level Design



#### **Project Directory Structure**

The directory structure is:

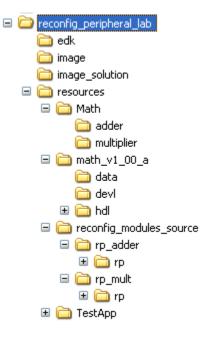


Figure 2: The Project Directory

- The edk\ directory is used to create a processor system.
- The resources \ directory contains:
  - Source files used to generate the netlists of the addition and multiplication functions,
  - A pre-compiled netlist for the addition and multiplication functions in Math and associate sub-directories, and
  - A software application to demonstrate the functionality.
- The math processor core (pcore) provides:
  - o The necessary processor bus connections
  - The required peripheral services (in this case, one slave register and a software reset), and
  - A placeholder for the math functionality module
- The image\ directory is used to hold the generated full configuration bitstream file in the System ACE<sup>™</sup> format and partial bitstream files.
- The image\_solution\ directory contains the final *system.ace* and partial bit files for a quick test.

# **Tutorial Steps**

This tutorial is separated into steps, followed by general instructions and supplementary detailed steps allowing you to make choices based on your skill level as you progress through the lab.

- Step 1: Creating a Processor Hardware System
- Step 2: Creating a Software Project
- Step 3: Creating a PlanAhead Project
- Step 4: Defining a Reconfigurable Partition
- Step 5: Adding Reconfigurable Modules
- Step 6: Defining the Reconfigurable Partition Region
- Step 7: Running the Design Rule Checker
- Step 8: Creating the First Configuration, Implementing, and Promoting
- Step 9: Creating Other Configurations, and Implementing
- Step 10: Running Partial Reconfiguration to Verify Utility
- Step 11: Generating Bit Files
- Step 12: Creating an Image, and Testing

## Step 1: Creating a Processor Hardware System

#### Creating a Processor System Using the Base System Builder (BSB) Wizard in XPS

- 1. Select Start > All Programs > Xilinx Design Tools > Xilinx Design Suite 14.1 > EDK > Xilinx Platform Studio to open XPS.
- 2. In the Getting Started page, click **Create New Project Using Base System Builder** to open a Create New XPS Project using BSB Wizard dialog box.
- 3. Browse to the reconfig\_peripheral\_lab\edk\ directory.
- 4. Click Save.
- 5. Keep the default options of using ISE tools and AXI System as the interconnect type, and click **OK**.

You will create a system for a Virtex®-6 ML605 evaluation platform.

- 1. In Board and System Selection form, select **Xilinx** as a Board Vendor.
- 2. In Board Name field, select Virtex-6 ML605 Evaluation Platform.
- 3. In Board Revision field, select **D**.
- 4. Click **Next** with other default options selected.
- 5. Select 50.00 MHz from the Processor Frequency drop-down menu.
- 6. Select 64 KB from the Local Memory Size drop-down menu.
- In the selected peripherals list on the right, remove all devices *except*: RS232\_Uart\_1 and SysACE\_CompactFlash.
- 8. Click **RS232\_Uart\_1** and configure it with a baud rate of **115200**.
- 9. Click Finish.
- 10. If the **Next Step** dialog box opens, click **OK** to start using Platform Studio and open the System Assembly View window as shown in the following figure.

<b>X</b> M M	Name	Bus Name	ID Trees	IP Version
IBB	Name	bus Name	IP Туре	IP version
	axi4lite_0		🙀 axi_interconnect	1.06.a
590-	microblaze_0_dlmb	<b>T</b> .	1mb_v10	2.00.b
	microblaze_0_ilmb	1mb_v10	2.00.b	
	microblaze_0	📩 microblaze	8.30.a	
▋▋▋レーレ	microblaze_0_bram_block		🙀 bram_block	1.00.a
	microblaze_0_d_bram_ctrl		1mb_bram_if_cntlr	3.00.b
	microblaze_0_i_bram_ctrl		☆ Imb_bram_if_cntlr	3.00.b
	🕀 debug_module		📩 mdm	2.00.b
<u> </u>	SysACE_CompactFlash		📩 axi_sysace	1.01.a
<u> </u>	H RS232_Uart_1		🙀 axi_uartlite	1.02.a
	clock_generator_0		t clock_generator	4.03.a
	proc_sys_reset_0		proc_sys_reset	3.00.a

Figure 3: Displaying the System Assembly View

#### Adding the Required IPs to the Processor System

Copy the <code>reconfig\_peripheral\_lab/resources/math\_v1\_00\_a/</code> folder to the <code>reconfig\_peripheral\_lab/edk/pcores/</code> folder.

#### **Partial Reconfiguration Design Details**

#### Examine the user\_logic.vhd file located in

reconfig\_peripheral\_lab\resources\math\_v1\_00\_a\hdl\vhdl\. It declares a component that will be used in reconfigurable partition at line 133. The same is instantiated at line 158. The data inputs to the component are clocked at lines starting at 191. The reset input to the component is a combination of the hardware bus reset and software reset. The software reset is generated by a soft\_reset block located at line 310 in math.vhd file located in the same directory. The software reset is necessary to reset the reconfigured logic after reconfiguring the partition.

Note: If line numbering is hidden from view in XPS, turn line numbers on as follows:

- 1. Select Edit > Preferences > ISE Text Editor.
- 2. Click to select the Show line numbers check box.
- 3. Click **Apply** and then **OK**.
- 4. Rescan the User Repositories in XPS by selecting **Project > Rescan User Repositories**.

In the IP Catalog tab, MATH displays in the **USER** folder under the Project Local pcores folder.

- 5. Expand the USER folder.
- 6. Select MATH.
- 7. Double-click **MATH** to add an instance of the IP to the System Assembly.



A properties form opens.

- 8. Click **OK** twice to add the IP with the default settings and connect it to the microblaze\_0 instance.
- 9. In the IP Catalog tab, select the FPGA Internal Configuration Access Port (v2.02.a) IP (axi hwicap) under the FPGA Reconfiguration folder, right-click and select **Add IP**.

This adds the instance of the IP to the System Assembly View.

10. Click **OK** twice to accept the default settings and connect the IP to the microblaze\_0 instance.

**Note:** When the IP cores are added, interface connections are made, and the addresses are automatically assigned.

#### **Connecting the Ports**

- 1. In the System Assembly View, select the **Ports** tab.
- 2. Expand the **axi\_hwicap\_0** instance.
- 3. Select Hardware > Launch Clock Wizard.
- 4. In the Clock Wizard form, select **50.0000** for the ICAP\_Clk of the axi\_hwicap\_0 instance, select **<AUTO>** under the source column, and click **OK**.
- 5. Click **OK** to close the form.

The connection appears as shown in Figure 4.



•	Bus Interfaces	Ports	Addresse	5			
N	Name		Con	nected Port	Direction	Range	Class
G	External Ports	******					
0	axi4lite_0						
E	microblaze_0_	dlmb					
0	microblaze_0_i	ilmb					
E	microblaze_0						
	microblaze_0_l	bram_bloc	:k				
E	microblaze_0_	d_bram_ct	trl				
6	microblaze_0_i	_bram_cti	rl				
E	debug_module	e					
E	axi_hwicap_0						
	-ICAP_Clk		cloc	c_generator_0::CLKOUT0			CLK
	EOS_IN				/ 1		
	IP2INTC_Ir	pt					INTERRUPT
	⊕ (BUS_IF) S	IXA	Con	nected to BUS axi4lite_0	-		
E	- SysACE_Comp	actFlash					
6	RS232_Uart_1						
6	math_0						
0	clock_generate	or_O					
E	proc_sys_reset	0					

#### Figure 4: Connecting Clock Source to ICAP

#### **Partial Reconfiguration Design Details**

The axi\_hwicap pcore allows a separate clock domain for the hwicap so it can be run at 100 MHz when the system is run at a higher speed. In this tutorial, the system clock is 50 MHz and hence, we are running the entire design in a single clock domain.

Notice that there is EOS\_IN port on the axi\_hwicap\_0 instance. This port is available for the designer to connect to a separate signal that can be asserted only when the system is stable and the reconfiguration can be done, to take care of a situation where reconfiguration command may be issued before the system is stable. You can instantiate a STARTUP block and connect the port correctly and automatically by selecting configuration parameter **instantiate STARTUP primitive in the HWICAP core** option in the hwicap pcore configuration GUI.

- 1. Select the **Bus Interfaces** tab.
- 2. Double-click on the axi\_hwicap\_0 instance and click on the check box of **instantiate STARTUP primitive in the HWICAP core** option in the User tab.
- 3. Click **OK** to accept the settings.
- 4. Select the **Ports** tab and observe that EOS\_IN port is not listed as it is connected to STARTUP block which is automatically instantiated with the selected option.



#### **Generating Netlists**

To run the Platform Generator, select **Hardware > Generate Netlist**.

This generates the peripheral and system netlists, and the system.bmm files, all of which are used during implementation in the PlanAhead tool.



## Step 2: Creating a Software Project

After the hardware netlist is generated, use the Software Development Kit (SDK) available with EDK to:

- Create a software project
- Import the provided source files
- Compile the provided source file
- Generate an executable file

# Exporting Hardware Design to SDK, and Creating a Board Support Package, making sure to add xilfatfs library support

- 1. In XPS, select **Project > Export Hardware Design** to SDK to launch SDK.
- 2. Uncheck Include bitstream and BMM file.
- 3. Click Export & Launch SDK.

A workspace location dialog box opens.

- 4. Browse to the reconfig\_peripheral\_lab\edk\SDK\SDK\_Export directory, and click **OK** to open SDK after importing hardware specification of the system.
- 5. In SDK, select File > New > Xilinx Board Support Package.

Notice that the default Project Name is **standalone\_bsp\_0** and the OS is **standalone**.

6. Click **Finish** with default settings.

The Board Support Package Settings window opens.

7. Check the **xilfatfs** check box to select the FAT file system support for the Compact Flash card.

Na	ame	Version	Description
lw	/ip140	1.01.a	IwIP TCP/IP Stack library: IwIP v1.4.0, Xilinx adapter v1.00.a
🗸 xil	lfatfs	1.00.a	Provides read/write routines to access files stored on a FAT16/32 file system
xil	lflash	3.01.a	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
📃 xil	lisf	2.04.a	Xilinx In-system and Serial Flash Library
xil	Imfs	1.00.a	Xilinx Memory File System

#### Figure 5: Selecting File System Support

8. Click **OK** to accept the settings and close the form.

# **E** XILINX.

#### **Creating a Xilinx C Project**

- 1. Select File > New > Xilinx C Project.
- 2. Type **TestApp** for the Project Name.
- 3. Select **Empty Application** in the Project Application Template pane.
- 4. Click Next.
- 5. Select **Target** an Existing Board Support Package.
- 6. Click Finish.

#### **Generating a Test Application**

- 1. In the Project Explorer view, select **TestApp**.
- 2. Right-click and select Import.
- 3. Double-click General.
- 4. Double-click File System.
- 5. Browse to the reconfig\_peripheral\_lab\resources\TestApp\src\ folder.
- 6. Click **OK**.
- 7. Select main.c and xhwicap\_parse.h.
- 8. Click Finish.

This compiles the source files and generates <code>TestApp.elf</code> in the <code>reconfig\_peripheral\_lab\edk\TestApp\Debug\</code> folder.



#### **Partial Reconfiguration Design Details**

Examine the reconfig\_peripheral\_lab\resources\TestApp\src\main.c file.

This code includes a function, beginning on line 164, which loads a partial bit file from the CompactFlash and writes to the ICAP.

The calls to this function, beginning on line 433, instruct the program to load a specific partial bit file and then assert software reset.

When the blank bitstream is loaded, the software reset is not required since there is no real logic residing in the reconfigurable region.

#### **Generating a Linker Script**

- 1. Be sure that the Heap and Stack sizes are set to 2048 (0x800).
- 2. In SDK Project Explorer view, select TestApp.
- 3. Right-click and select Generate linker script.
- 4. Change the Heap size and the Stack size to **2048**.

Generate a linker script			-	the local division in	X
Generate linker script Control your application's mer	mory map.				
Project: TestApp Output Script: peripheral_lab\edk\SDK\SDK Modify project build settings Set generated script on all pr Memory microblaze_0_i_bram_ctrl	as follows: oject build conf Base Address	igurations Size	Browse	Basic Advanced Place Code Sections in: Place Data Sections in: Place Heap and Stack in: Heap Size: Stack Size:	microblaze_0_i_bram_ctrl_microblaze_0_d_bram_ctrl  microblaze_0_i_bram_ctrl_microblaze_0_d_bram_ctrl microblaze_0_i_bram_ctrl_microblaze_0_d_bram_ctrl KB
?					Generate Cancel

#### Figure 6: Generating a Linker Script

- 5. Click Generate.
- 6. Click **Yes** to overwrite the existing copy and recompile the application again.
- 7. Select File > Exit to close SDK.
- 8. In XPS, select **File > Exit** to close XPS.



# **Step 3: Creating a PlanAhead Project**

Now that you have generated the required netlist files for the design, you will use the PlanAhead tool to:

- Floorplan the design
- Define reconfigurable partitions
- Add reconfigurable modules
- Run the implementation tools
- Generate full and partial bitstreams

In this step, you will create a new project.

#### **Creating a PlanAhead Project, and Importing the Generated Netlist** Files

- To open PlanAhead, select Start > All Programs > Xilinx Design Tools > Xilinx ISE Design Suite > PlanAhead > PlanAhead.
- 2. Click Create New Project.
- 3. Click Next.
- 4. Browse to and select the reconfig\_peripheral\_lab\ directory for the Project location.
- 5. Click Select.
- 6. Type **PlanAhead** for the Project name in the New Project wizard.
- 7. Make sure that Create Project Subdirectory is checked.

Project Name Enter a nam will be stored	e for your project and specify a directory where the project data files	¢
Project name:	PlanAhead	0
Project location:	C:\PartialReconfiguration\reconfig_peripheral_lab	
Create Proje	ct Subdirectory eated at: C:\γeconfig_peripheral_lab\PlanAhead	
	< Back Next > Finish	Cancel

Figure 7: Project Name Page of the New Project Wizard



- 8. Click Next.
- 9. In the Project Type page, select Post-synthesis Project.
- 10. Check the Enable Partial Reconfiguration option.

**Note:** If you forget to check the option, you can still enable it from the project (netlist based only) by selecting **Tools > Project Settings > General** and clicking the **Partial Reconfiguration Project** check box. This must be done before a partition can be defined as reconfigurable.

11. Click Next.

**Important!** The Enable Partial Reconfiguration option is available only if you have a license for Partial Reconfiguration.

New Pro	oject	X
Project 1 Specify	Type y the type of project to create.	<b>R</b>
You w and a	Project vill be able to add sources, generate IP, run RTL analysis, synthesis, implementation, design plann analysis. o not specify sources at this time	ing
You w	synthesis Project vill be able to add sources, view device resources, run design analysis, planning and implementatio o not specify sources at this time nable Partial Reconfiguration	m.
	lanning Project ot specify design sources. You will be able to view part/package resources.	
	rt I <u>S</u> E Place & Route results vill be able to do post-implementation analysis of your design.	
	rted Project te a PlanAhead project from a Synplify, XST or ISE Project File.	
	< Back Next > Finish Car	ncel

**Figure 8: Importing Synthesized Netlists** 

- 12. Click the Add Files button.
- 13. Browse to the reconfig\_peripheral\_lab\edk\implementation\ folder.
- 14. Select all NGC files including the system.ngc file, and click **OK**.



15. In the Top column, click the radio button next to system.ngc to identify it as the top-level design file.

	Id	Name	Тор	Location	
6	1	system.ngc	۲	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
3	2	system_axi4lite_0_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
1	3	system_axi_hwicap_0_wrapper.ngc	Õ	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
7	4	system_axi_hwicap_0_wrapper_fifo_generator_v8_3_1.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
	5	system_axi_hwicap_0_wrapper_fifo_generator_v8_3_2.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
1	6	system_clock_generator_0_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	-
	7	system_debug_module_wrapper.ngc	0	C: \PartialReconfiguration \reconfig_peripheral_lab \edk \implementation	X
1	8	system_math_0_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
	9	system_microblaze_0_bram_block_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	1
	10	system_microblaze_0_d_bram_ctrl_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	-
3	11	system_microblaze_0_dlmb_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	÷
	12	system_microblaze_0_i_bram_ctrl_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	-
	13	system_microblaze_0_ilmb_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
-	14	system_microblaze_0_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
_	15	system_proc_sys_reset_0_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
		system_rs232_uart_1_wrapper.ngc	0	C:\PartialReconfiguration\reconfig_peripheral_lab\edk\implementation	
	17	system_sysace_compactflash_wrapper.ngc	0	$C: \label{eq:configuration} \end{tabular} experimentation t$	
		Add Files		Add Directories	

Figure 9: Selecting the Top-level Netlist File

16. Click Next.

The Add Constraint files (optional) page opens.

- 17. Click Add Files.
- 18. Browse to reconfig\_peripheral\_lab\edk\data\
- 19. Select system.ucf.
- 20. Click **OK**.
- 21. Click **Next** to open the Product Family and Default Part page.
- 22. Make sure that the **xc6vlx240tff1156-1** part is selected. Otherwise, select the filters, and select the **xc6vlx240tff1156-1** part as shown in the following figure.



Specify	Filter							
🛞 Parts	Product Ca	tegory Gene	ral Purpose	-	Package	FF1156		-
Boards		Family Virte		•	Speed Grade			*
	Sub	-Family Virte	x-6 LXT	*	Temp Grade	С		-
Search: Q								
Device		I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Trar
xc6vlx13		1,156	600	80000	160000	264	480	20
		1,156	600	124800	249600	344	640	20
xc6vlx19		-			_			
xc6vlx19 xc6vlx24 xc6vlx36	Otff1156-1	1,156	600 600	150720 227520	301440 455040	416 416	768 576	20 20

#### Figure 10: Selecting the Target Device

#### 23. Click Next.

#### 24. Click Finish.

The project is created. The Project Manager pane displays the modules present in the design.

#### **E** XILINX.

ources	_ O @ ×
N 🖾 😂 📾 🔂 🛃	
Design Sources (17)	
□ ··· ·· ·· ·· ·· ·· ·· ·· ·· ·· ·· ·· ·	
• system_axi4lite_0_wrapper.ngc	
• system_axi hwicap 0 wrapper	
• system_axi_hwicap_0_wrapper	
• system_axi_hwicap_0_wrapper	
• system_dock_generator_0_wra	
• system_debug_module_wrappe	
• system_math_0_wrapper.ngc	
• system_microblaze_0_bram_blo	ck wrapper.ngc
• system_microblaze_0_d_bram_d	
system_microblaze_0_dlmb_wra	apper.ngc
system_microblaze_0_i_bram_c	trl_wrapper.ngc
system_microblaze_0_ilmb_wrap	pper.ngc
system_microblaze_0_wrapper.	ngc
system_proc_sys_reset_0_wrap	pper.ngc
system_rs232_uart_1_wrapper.	.ngc
system_sysace_compactflash_v	wrapper.ngc
Constraints (1)	
in constrs_1	
system.ucf (target)	

Figure 11: Design Hierarchy in PlanAhead

#### **Step 4: Defining a Reconfigurable Partition**

This design has one reconfigurable partition that you must explicitly define.

#### Defining a Reconfigurable Partition (RP) With a Black Box Reconfigurable Module (RM).

1. Click **Open Synthesized Design** under **Netlist Analysis** step of the Project Manager Flow Navigator pane to invoke the netlist files parser.

This is necessary to access a lower-level module to define a reconfigurable partition.

A warning message indicating that one instance will be converted to a black box because the netlist file for it is missing. This is expected because no netlist has been associated with this module yet.

A Netlist tab displays the hierarchical view of the system.

- 2. Click **OK** twice.
- 3. Expand the math\_0 instance.
- 4. Select math\_0/USER\_LOGIC\_I/rp\_instance in the Netlist view.

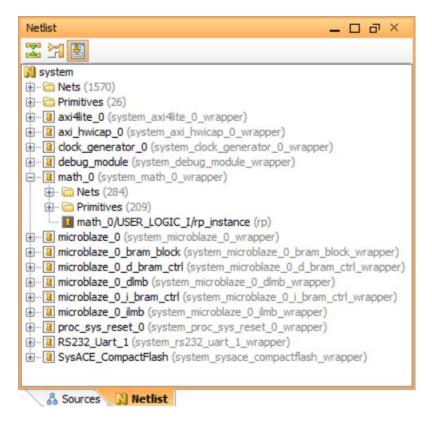


Figure 12: Netlists Hierarchy View



5. Right-click and select **Set Partition**.

The Set Partition dialog box opens.

- 6. Click **Next** *twice*.
- 7. Select Add this Reconfigurable module as a black box without a netlist.
- 8. Type **math\_BB** in the RM name field since the partition does not yet have a defined netlist.

Set Partition
Reconfigurable Module Name Enter a name for the new Reconfigurable Module for instance 'math_0/math_0/USER_LOGIC_I/rp_instance'.
Reconfigurable Module Name:       math_BB         Netlist already available for this Reconfigurable Module         Add this Reconfigurable Module as a black box without a netlist
< Back Next > Finish Cancel

Figure 13: Setting a Partition

- 9. Click Next.
- 10. Click Finish.

**Note:** The black box icon has changed to a diamond shape.

## **Step 5: Adding Reconfigurable Modules**

This design has two Reconfigurable Modules (RMs) for the Reconfigurable Partition (RP). In this step, you add the two modules.

#### Adding Two Reconfigurable Modules: Adder and Multiplier

- 1. In the Netlist window, select the math\_0/USER\_LOGIC\_I/rp\_instance.
- 2. Right-click and select Add Reconfigurable Module.

The Add Reconfigurable Module dialog box displays.

- 3. Click Next.
- 4. In the Reconfigurable Module Name field, type **adder**.
- 5. Verify that Netlist already available for this Reconfigurable Module is selected.

Add Reconfigurable Module
Reconfigurable Module Name
Enter a name for the new Reconfigurable Module for instance 'math_0/math_0/USER_LOGIC_I/rp_instance'.
Reconfigurable Module Name: adder
Netlist already available for this Reconfigurable Module
Add this Reconfigurable Module as a black box without a netlist
< Back Next > Finish Cancel

#### Figure 14: Adding a Reconfigurable Module

6. Click **Next**.

Browse to reconfig\_peripheral\_lab\resources\Math\adder\ and select the rp.ngc file.

7. Click **OK**.



Add Reconfigurable Mo	dule	×
Specify Top Netlist File Specify the EDIF or NGC	e netlist that contains the Reconfigurable Module	¢
Top Netlist File: C:\Partial Netlist directories (option	Reconfigurationlyeconfig_peripheral_lablyesources\Math\adderlyp.ngc	
Copy Sources into Pro	Add Directories	
	< Back Next > Finish	Cancel

Figure 15: Locating the adder Version of math.ngc

- 8. Click Next twice.
- 9. Click Finish.

In the Netlist pane, expand Reconfigurable Modules hierarchy under math\_0/USER\_LOGIC\_I/rp\_instance to view the adder RM entry.

#### 10. Follow the steps in Step 5 to add a multiplier RM from the

 $\lable reconfig_peripheral_lab\resource\Math\multiplier\rp.ngc\ directory.\ Name the RM\ mult.$ 

The Netlist window displays three Reconfigurable Modules (including the black box) for the math Reconfigurable Partition.



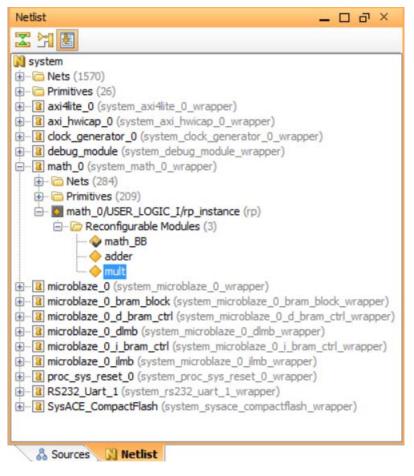


Figure 16: PlanAhead Project with adder and mult RMs added

# **Step 6: Defining the Reconfigurable Partition Region**

Next, floorplan the RP region. Depending on the type and amount of resources used by each RM, the RP region must be appropriately defined so it can accommodate any RM variant.

#### Setting the Reconfigurable Region

- 1. Zoom to the top left quarter of the FPGA.
- 2. Select Window > Physical Constraints.
- 3. In the Physical Constraints tab, select pblock\_math\_0/USER\_LOGIC\_I/rp\_instance.
- 4. Right-click, and select **Set Pblock Size**.

Physical Constraints		_ [	5 67	×
🔍 🛣 🖨 🖪				
😬 netlist_1				
- P ROOT				
💽 pblock_math_0	USER_LO	GIC_I_rp	_insta	nce
		-		
Pblock Properties			50	

**Figure 17: Setting Physical Constraints** 

- 5. Move the cursor in the Device window.
- 6. Click and drag the cursor to draw a box that bounds SLICE\_X8Y230:SLICE\_X17Y239, as shown in the next figure.

You must draw a box around this region because the multiplier (mult) RM requires one DSP48E and the adder RM requires 32-bit tall carry chain.

The current grid coordinates are reported in the status bar at the bottom of the PlanAhead window.

At the completion of this step, the Set Pblock dialog box displays.

∑ Project Summary × 🚱 Device × 🔄 Schematic ×	07	×
		4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
		1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
		AAA
		0000
		4444
		4444
		1111

Figure 18: Closer View of the Pblock Area

- 7. In the Set Pblock dialog box, verify that **SLICE** and **DSP48** are checked as the resources to be reconfigured, shown in the following figure.
- 8. Click **OK**.



irids		
SLICE		
USP 10		

Figure 19: Setting Pblock with SLICE and DSP48



## **Step 7: Running the Design Rule Checker**

Xilinx recommends that you run a Design Rule Check (DRC) in order to detect errors as soon as possible.

#### Selecting and Running PR-specific DRCs

- 1. Select **Tools > Report DRC**.
- 2. Deselect All Rules.
- 3. Select Partial Reconfig.
- 4. Click **OK** to run the PR-specific design rules.

	drc_1		
Output File:			
Rules to Che	ck: 14 of 1229		
् 🖨 🛣			
All Ru	es (1229)		
🖶 🔲 d			
🕀 🔲 Ba	ank (19)		
🕀 🔲 IC	<b>B</b> (15)		
🕀 🔲 Pl	acer (14)		
🕀 🔲 Ne	etlist (1139)		
T	oorplan (5)		
🕂 🔲 C			
⊕			
E R	a contract of the second s		
I.	artition (10)		
1	artial Reconfig (14)		
	eam Design (3)		
	Select All	Clear All	

Figure 20: Running Design Rule Checks

You will see warnings stating that Reconfigurable Modules (RMs) have not been implemented.



# Step 8: Creating the First Configuration, Implementing, and Promoting

Now you can create and implement the first configuration.

#### **Creating a New Strategy**

Use the **-bm** option pointing to the system.bmm file for the new strategy.

- 1. Select **Tools > Options**.
- 2. Select **Strategies** in the left pane.
- 3. Select ISE 14 in the Flow drop-down box.
- 4. Under PlanAhead Strategies, select ISE Defaults.
- 5. Click the + button to create a new strategy.

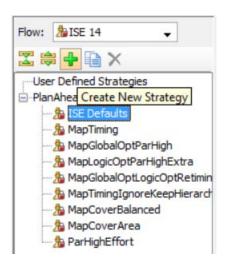


Figure 21: Creating a New Strategy

- 6. Name the new strategy **ISE14\_BM**, and set the Type to **Implement**.
- 7. Click **OK**.
- 8. Under Translate (ngdbuild), click in the More Options field.
- 9. Type -bm ..\..\edk\implementation\system.bmm, and click Apply.
- 10. Click **OK**.



#### **Running the Implementation Using Mult as a Variant**

- 1. At the bottom of the PlanAhead tool user interface, select the **Design Runs** tab.
- 2. Select the **config\_1** run.
- 3. In the Implementation Run Properties window, select the **General** tab.
- 4. In the Name field, type **mult** as the run name.
- 5. Click **Apply** to change the run name from config\_1 to **mult**.

← → 😳 > mult	
Name:	mult
Part:	xc6vlx240tff1156-1 (active)
Description:	ISE Defaults, including packing regi:
Status:	Not started
Constraints:	📅 constrs_1 (active) 🔻
4	

Figure 22: Implementation Run Properties View

- 6. In the Options tab, change the Strategy to **ISE14\_BM**.
- 7. Click Apply.
- 8. In the Partitions tab, click the Module Variant column drop-down button and select **mult** as the variant.
- 9. Click **Apply**.



► → 💁 k • mult			
Name	Module Variant	Action	Import
Static Logic		Implement -	N/A
math_0/math_0/USER_LOGIC_I/rp_instance	math_BB 🚽	Implement +	N/A
	math_BB adder mult		
		<u>,</u>	

Figure 23: Implementation Run

- 10. In the Design Run window, select **mult**, and right-click and select **Launch Runs** to run the implementation.
- 11. Select Launch Runs on Local Host.
- 12. Click **OK**.
- 13. Click **Save** to save the project and run the implementation.

The implementation runs.

When implementation is finished running, a dialog box opens in which you can load the implemented results, or promote the implemented partitions, among other options.

- 14. Select the **Promote Partitions** radio button, and click **OK**.
- 15. In the Promote Partitions dialog box, click **OK** to promote the current configuration so the implemented results are available for the subsequent configurations.



runs.	promote directory. After promoting runs, you can import the partitions into
elect Runs to p 🛣 🚔	romote
Run	Directory
mult	onfig_peripheral_lab\PlanAhead\PlanAhead.promote\Xmult
and the second sec	_0/math_0/USER_LOGIC_I/rp_instance - mult
	Select Implemented Clear All

**Figure 24: Promoting Partitions** 



# Step 9: Creating Other Configurations, and Implementing

After you have created the first configuration, the static logic implementation is reused for the rest of the configurations. Next, you will create the desired number of additional configurations and implement them.

#### **Creating Multiple Runs**

1. Select Flow > Create Runs.

The Create New Runs window opens.

- 2. Click Next twice.
- 3. In the Choose Implementation Strategies and Reconfigurable Modules page, change the name of the configuration from config\_1 to **adder**.
- 4. Click More.
- 5. Change the name of config\_1 to **black\_box**.

C	Create New R	uns		X
		ementation Strategies and Rec onfigure runs using various strategies		figurable
	Create Impleme	ntation Runs		
	Name	Strategy	Make Active (optional)	Partition Action
	adder	👗 ISE14 BM (ISE 14)	۲	math_0/m; 🗙
	black_box	🗏 ISE14 BM (ISE 14)	©	math_0/m; 🖳 🗙
	More			Runs to create: 2
			< Back Next >	Finish

Figure 25: Creating Multiple Runs

6. In the adder configuration row, click the **Partition Action** field.



7. For the rp\_instance row, click the Module Variant column drop-down arrow, and select **adder** as the variant to be implemented, as shown in Figure 26.

Specify Partition		1	
Choose Module Variants for Reconfigura	able Modules and a	ctions	
Name	Module Variant	Action	Import from
Static Logic		Import	← C:\PartialReconfiguration\reconfig_peripheral_lab\PlanAhe
Math_0/math_0/USER_LOGIC_I/rp_instance	mult 🚽	Import	- C:\PartialReconfiguration\reconfig_peripheral_lab\PlanAhe
	math_BB		
	adder mult		
	marc	1	
			OK Cancel

#### Figure 26: Selecting adder Module Variant

- 8. Click **OK**.
- 9. Similarly, select math\_BB variant for the black\_box run (row).
- 10. Click Next.
- 11. Select Launch Runs on Local Host.
- 12. Click Next.
- 13. Click **Finish** to run the implementations for both configurations.
- 14. Click **Cancel** when the runs are finished.



# **Step 10: Running Partial Reconfiguration to Verify Utility**

Next, you will check to be sure that the static implementation, including interfaces to reconfigurable regions, is consistent across all configurations. To verify this, you can run the PR\_Verify utility.

#### Running the PR\_Verify Utility

- 1. Run the PR\_Verify utility to make sure that there are no errors.
- 2. In the Configurations window, select any of the configurations.
- 3. Right-click, and select Verify Configuration.

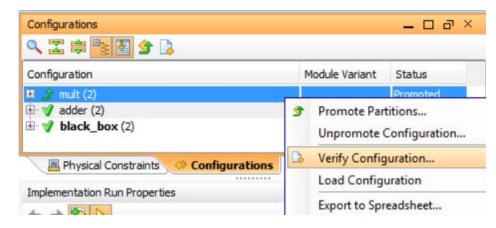


Figure 27: Verifying All Configurations

- 4. Press **Shift** and select all configurations.
- 5. Click **OK**.
- 6. The PR\_Verify utility runs and reports that there were no errors.

# **Step 11: Generating Bit Files**

After PR\_Verify validates all the configurations, you can generate full and partial bit files for the entire project.

#### **Generating Full and Partial Bitstreams**

- 1. In the Design Runs window, press **Shift** and select the following three designs runs:
  - mult
  - adder
  - black\_box
- 2. Right-click and select Generate Bitstream.

This runs the bitstream generation process and generates full and partial bitstreams.

The bit files are placed in the mult, adder and black box directories under the reconfig\_peripheral\_lab\PlanAhead\PlanAhead.runs\ directory.

- 3. Click **OK**.
- 4. Save the project.
- 5. Close PlanAhead by selecting **File > Exit**.

## Step 12: Creating an Image, and Testing

For this step you need to open an EDK shell, and create both a download.bit and a system.ace file in the image\ directory. Copy the generated partial bit files, place them in the image\ directory, and name them adder.bit, mult.bit, and blank.bit.

#### Renaming Partial Bitstream Files, and Generating the system.ace File

 Launch the ISE Design Suite command prompt from your Windows environment by selecting Start > All Programs > Xilinx Design Tools > Xilinx ISE Design Suite > Accessories > ISE Design Suite Command Prompt.

- 2. In the command window, go to the reconfig\_peripheral\_lab\image\ directory.
- 3. Execute the following command to generate the download.bit file (with the software component included) from adder.bit (with the hardware component) only.

```
data2mem -bm ..\edk\implementation\system_bd
-bt ..\PlanAhead\PlanAhead.runs\adder\adder.bit
-bd ..\edk\SDK\SDK_Export\TestApp\Debug\TestApp.elf tag microblaze_0 -o b
download.bit
```

**Hint:** Copy the command text from this document and paste it in the shell or command window by right-clicking and selecting **Paste**.

This generates the download.bit in the image \ directory.

4. In the Bash shell, execute the following command to generate the system.ace file in the image\ directory.

xmd -tcl genace.tcl -jprog -target mdm -hw download.bit -board ml605 -ace system.ace

5. Using Windows Explorer, copy and rename the following files, as shown in the following table.



File Name	Copy to Directory	Rename File To
reconfig_peripheral_lab\PlanAhead\PlanAhead.run s\adder\ adder_math_0_math_0_user_logic_i_rp_instance_ad der_partial.bit	\image	adder.bit
reconfig_peripheral_lab\PlanAhead\PlanAhead.run s\mult\mult_math_0_math_0_user_logic_i_rp_insta nce_mult_partial.bit	\image	mult.bit
reconfig_peripheral_lab\PlanAhead\PlanAhead.run s\black_box\black_box_math_0_math_0_user_logic _i_rp_instance_math_bb_partial.bit	\image	blank.bit

#### Copying the system.ace and Three Partial Bit Files on a Compact Flash Memory Card

- 1. Place a blank Compact Flash memory card in a Compact Flash writer.
- 2. Using Windows Explorer, copy the three partial bit files and the system.ace file from reconfig\_peripheral\_lab\image\ folder to the Compact Flash card.
- 3. Place the Compact Flash card in the ML605 board.
- 4. Set the SACE Mode pins (S1) to **0111 (dn-up-up-up)** to configure the FPGA device from the Compact Flash.
- 5. Connect your PC to the ML605 with the provided USB cable.
- Install the driver, if necessary. For instructions, see the ML605 Hardware User Guide: <u>http://www.xilinx.com/support/documentation/boards\_and\_kits/ug534.pdf</u>
- 7. Start a HyperTerminal window, connecting using **COMx at 115200 baud** and power **ON** the ML605 board.
- 8. Press CPU Reset.
- 9. Follow the menu and test various reconfigurations.

# Conclusion

In this tutorial, you:

- Created a processor system using XPS.
- Added a user peripheral which included a place holder for the reconfigurable partition.
- Generated netlist files.
- Created an application using SDK.
- Generated a full bitstream as well as partial reconfiguration bitstreams using the PlanAhead tool.
- Generated an ACE file for Compact Flash memory card.
- Verified the functionality using the ML605 evaluation board.